A Machine Learning Approach to Cyberbullying Detection

---

1. Introduction

Cyberbullying is a prevalent and growing problem in today's digitally connected world, particularly among younger demographics on social media platforms. Detecting harmful content automatically using machine learning is an essential step in moderating such interactions and fostering safe online spaces. In this challenge, we address the problem using Natural Language Processing (NLP) and classification models, ultimately selecting XGBoost with SMOTE oversampling to address class imbalance.

---

2. Data Preprocessing and Feature Engineering

2.1 Dataset Overview

The input dataset was structured as a list of labeled text entries in the format:

"sample text": True/False

These entries indicate whether the text contains cyberbullying (True) or not (False). Upon parsing and validation, I constructed a structured Data Frame with the fields:

- text: the original message
- label: a Boolean indicating cyberbullying (True) or not (False)

2.2 Text Parsing and Cleaning

To extract the data correctly, a regular expression was used:

r"""["']?(.*?)["']?:\s*(True|False),?$"""

This ensured reliable separation of messages from their labels.

2.3 NLP Preprocessing Pipeline

The following steps were taken to prepare the text:

- **Lowercasing:** To normalize input.
- **Punctuation Removal:** All punctuation was stripped using regex.
- **Tokenization:** Text was split into individual words.
- **Stopword Removal:** Using NLTK's stopwords.words('english').
- **Lemmatization:** Each word was reduced to its base form using WordNetLemmatizer.

Example:

Raw: "You're such a loser!!! Nobody likes you."

Preprocessed: "loser nobody like"

2.4 Feature Vectorization

TF-IDF (Term Frequency–Inverse Document Frequency) was used to convert text into numerical vectors. The TfidfVectorizer was restricted to the top 3000 most informative features.

vectorizer = TfidfVectorizer(max_features=3000)

X = vectorizer.fit_transform(clean_df["clean_text"])

---

3. Model Architecture and Training

3.1 Handling Class Imbalance

The dataset exhibited significant class imbalance (i.e., fewer True than False examples). To combat this, **SMOTE (Synthetic Minority Oversampling Technique)** was applied to generate synthetic samples for the minority class in the training set.

smote = SMOTE(random_state=42)

X_resampled, y_resampled = smote.fit_resample(X_train, y_train)

3.2 Model Choice: XGBoost

We chose **XGBoost (Extreme Gradient Boosting)** due to its:

- Robust performance on imbalanced datasets.
- Resistance to overfitting.

- Compatibility with sparse input (like TF-IDF).

model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')

model.fit(X_resampled, y_resampled)

3.3 Threshold Adjustment

Instead of using the default classification threshold of 0.5, we manually adjusted it to **0.4** to better capture minority class cases (cyberbullying instances). This decision was informed by evaluating precision-recall tradeoffs.

---

4. Results and Evaluation

The model was evaluated on an untouched test set (20% split). Below is a summary of the performance:

4.1 Classification Report (Threshold = 0.4)

	precision	recall	f1-score	support
False	0.90	0.89	0.90	113
True	0.48	0.50	0.49	22
accuracy			0.83	135
macro avg	0.69	0.70	0.69	135
weighted avg	0.83	0.83	0.83	135

4.2 Strengths

- **High overall accuracy (83%)**
- **Improved recall for minority class** (0.50 after threshold tuning)
- **Effective feature extraction with TF-IDF**

4.3 Weaknesses

- Lower precision and recall for the cyberbullying class (True), indicating some misclassifications.

- SMOTE might introduce synthetic noise, potentially affecting generalizability.

---

5. Insights and Future Directions

5.1 Key Learnings

- Class imbalance significantly impacts detection performance, and oversampling techniques like SMOTE can help.
- Lowering the decision threshold allows more positive class cases to be captured, though at the cost of precision.
- XGBoost is a strong baseline model, especially when paired with engineered features.

5.2 Potential Improvements

- Use **BERT or transformer-based models** (e.g., distilBERT) to capture contextual word meaning.
- Implement **hyperparameter tuning** using GridSearchCV or Optuna.
- Incorporate **more linguistic features** like sentiment, emotion, or profanity scores.
- Explore **ensemble learning** with voting classifiers to boost recall.

---

6. Conclusion

This project demonstrated a practical application of text classification for cyberbullying detection. By leveraging TF-IDF features, SMOTE for class balancing, and the power of XGBoost, we achieved reasonable detection accuracy. However, further work is needed to enhance the minority class performance, potentially using deep learning methods and more contextual embeddings.
